C H I L D 8

G. Becattini - F. Pirri

R P N / 8

G. Becattini - S.Giusti - F.Pirri

Ristampa dei rapporti interni R4-76 ed
R6-76 dell'istituto di Elettronica della
Facolta' di Ingegneria di Firenze.

general processor

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA MONTEBELLO, 3 - 3a rosso
TEL. 055 / 219.143 — 50123 FIRENZE

C H I L D 8

G. Becattini - F. Pirri

R P N / 8

G. Becattini - S.Giusti - F.Pirri

Ristampa dei rapporti interni R4-76 ed R6-76 dell'istituto di Elettronica della Facolta' di Ingegneria di Firenze.

GP general processor

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA MONTEBELLO, 3 - 3a rosso
TEL. 055 / 219.143 – 50123 FIRENZE

R 6 - 76

RPN/8

Uno studio sperimentale per
un linguaggio di programmazione

G. BECATTINI [x] - S. GIUSTI [xx] - F. PIRRI [xx]

x G. BECATTINI - GENERAL PROCESSOR - Firenze

xx S. GIUSTI; F. PIRRI - Università degli Studi di Firenze - Facoltà di Ingegneria - Istituto di Elettronica - Via di S. Marta, 3 - Firenze

1 - Scopi e finalita' del linguaggio RPN/8

Il linguaggio RPN/8 e' stato creato allo scopo di inquadrare le problematiche connesse con lo studio di un interprete ad alto livello destinato a funzionare sui microcomputer ad 8 bit della serie CHILD.

Una serie di motivi di ordine pratico ha suggerito di rendere possibile il funzionamento dell'RPN/8 anche su macchine aventi una configurazione di memoria di un kilo byte; in vista di questa limitazione e delle finalita' di questo studio non si e' ritenuto conveniente dotare l'interprete di certe prerogative tipiche dei comuni linguaggi ad alto livello.

In particolare le limitazioni piu' pesanti riguardano la capacita' aritmetica ridotta alla operazione di addizione fra numeri interi compresi tra 0 e 99. Inoltre le possibilita' di editing sono molto limitate.

Queste limitazioni, come gia' detto, possono essere considerate inessenziali in quanto gia' in diverse parti della forma attuale, non definitiva, l'RPN/8 opera secondo principi sufficientemente generali.

2 - Sviluppi dell'RPN/8

In base alle esperienze acquisite il linguaggio definitivo che verra' progettato ricalchera' in molti aspetti lo schema generale dell'RPN/8.

Oltre ovviamente a provvedere una aritmetica piu' potente ed un edit piu' sofisticato si tendera' a renderne piu' facile l'utilizzo ed in particolare ad abbinare alle capacita' di una macchina tradizionale, i vantaggi di una struttura a microprocessore (1,2,3,4).

3 - Struttura dell'interprete RPN/8

L'interprete e' caratterizzato da due modi di funzionamento: il modo di INPUT in cui si introduce o si modifica il programma sorgente, ed il modo di EXEC in cui si esegue il programma memorizzato.

Corrispondentemente a questa dualita' di operazione si possono in prima approssimazione riconoscere due parti fondamentali nella architettura del sistema. Poiche' tali parti non sono spesso fisicamente distinte, e' opportuno esaminare piu' nel dettaglio i vari blocchi logici che compongono l'RPN/8.

3.1 - Il modo di INPUT

In fase di INPUT (la macchina stampa ":") ogni carattere battuto alla tastiera viene esaminato e, in base alla sua codifica ASCII, si decide se introdurlo in memoria come facente parte del programma o se (carattere "↑") passare il controllo all'unita' di decodifica comandi. Una terza eventualita' e' costituita dal carattere RUBOUT (eco " ") che serve per la cancellazione dalla memoria dello ultimo carattere introdotto (Fig. 3.1.1) .

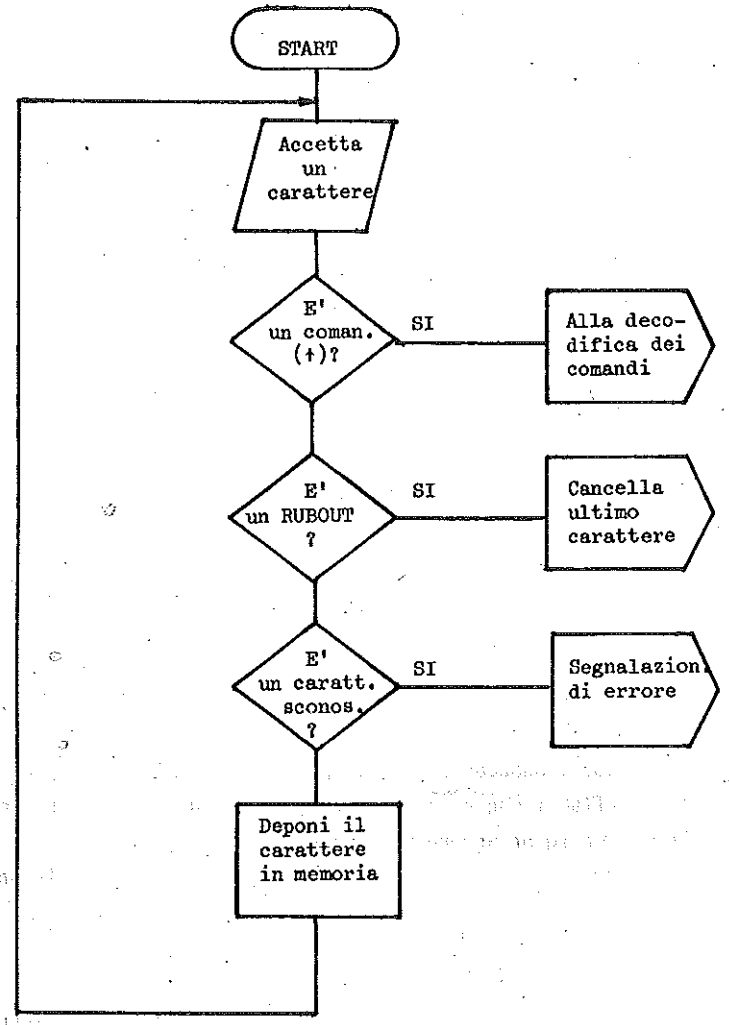

START
Accetta un carattere
E' un coman. (+)?
SI
Alla decodifica dei comandi
E' un RUBOUT ?
SI
Cancella ultimo carattere
E' un caratt. sconos. ?
SI
Segnalazion di errore
Deponi il carattere in memoria

Figura 3.1.1

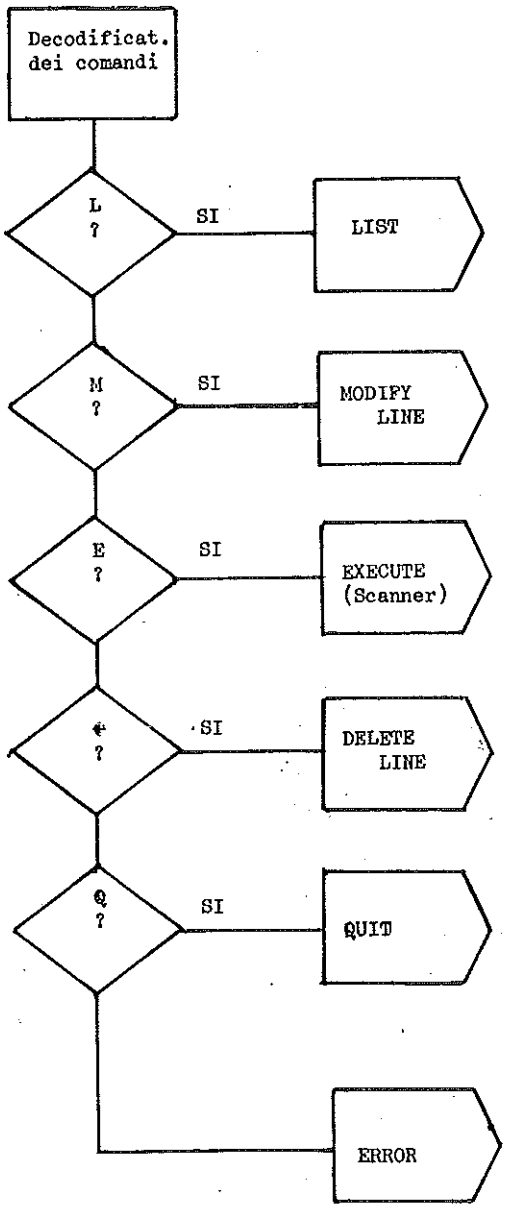

Decodificat.
dei comandi
L
?
SI
LIST
M
?
SI
MODIFY
LINE
E
?
SI
EXECUTE
(Scanner)
D
?
SI
DELETE
LINE
Q
?
SI
QUIT
ERROR

Figura 3.1.2

Il decodificatore dei comandi (richiamato come gia' detto dal carattere "↑") controlla se l'operazione richiesta appartiene all'insieme di quelle riconosciute ed eseguibili dall'interprete eventualmente segnalando la condizione di errore e passa il controllo alle opportune procedure (Fig. 3.1.2). Uno dei possibili comandi e' quello che serve per passare nella fase di EXEC.

3.2 - Il modo di EXEC

In conseguenza al comando ↑E il controllo viene passato al' l'analizzatore lessicale che in base alla prima parte della codifica ASCII di ogni carattere del programma (vedi tabella 3.2.1) stabilisce

Codifica ASCII	Significato
2x	Operatore
3x	Numero
4x	Variabile
5x	Etichetta
ØD	EOR (Car.ret.)

Tabella 3.2.1

la routine da richiamare, sospendendo temporaneamente la sua analisi (Fig. 3.2.1).

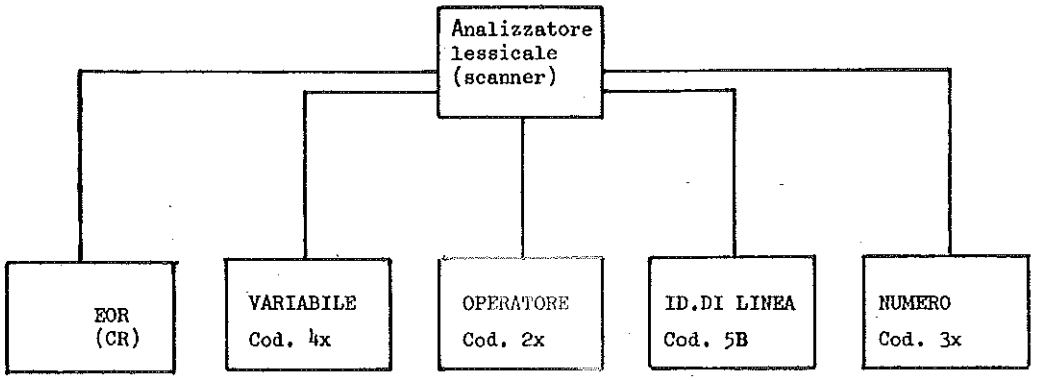

Analizzatore
lessicale
(scanner)
EOR
(CR)
VARIABILE
Cod. 4x
OPERATORE
Cod. 2x
ID.DI LINEA
Cod. 5B
NUMERO
Cod. 3x

Figura 3.2.1

4 - Descrizione operativa dell'RPN/8

Una volta caricato l'RPN/8 in memoria (inizio esecuzione loc. ØØØØ) si preme RETURN e la macchina stampa

```
* R P N / 8
:
```

Il carattere ":" avverte che ci si trova nello stato di INPUT e che l'operatore puo' controllare la macchina.

Un programma e' costituito da una o piu' linee (records) di lunghezza variabile. Ogni linea puo' essere identificata da una etichetta costituita da un numero di una o due cifre racchiuse tra parentesi quadre. La numerazione delle linee non deve necessariamente corrispondere all'ordine logico di esecuzione. Pertanto una linea alla quale non si faccia mai riferimento all'interno del programma puo' anche non essere provvista di identificatore. Si noti comunque che in fase di edit non e' possibile richiamare per la correzione frasi non etichettate.

Ogni linea termina con un carattere "end of record" (ritorno carrello). Un programma deve terminare con il carattere "end of file" (%).

4.1 - Comandi di sistema

Ogni comando di sistema deve essere preceduto dal carattere "↑" che non produce eco sullo stampato.

4.1.1 - Comando C (clear)

Serve per cancellare il programma dalla memoria; le variabili memorizzate precedentemente al comando C non risultano alterate.

4.1.2 - Comando E (execute)

Permette di passare nel modo di EXEC eseguendo il programma precedentemente introdotto. Se il programma termina normalmente o con una condizione di errore il controllo viene restituito all'operatore (stampa ":"). Diversamente (p.e. nel caso di un ciclo ripetuto indefinitamente) e' necessario premere il pulsante "reset" sul pannello del microelaboratore ed il tasto RETURN; la telescrivente stampa nuovamente l'intestazione.

4.1.3 - Comando L (list)

Con il comando L si ottiene la stampa del programma memorizzato. Il carattere EOF (%) non viene ristampato.

ESEMPIO:

```
:L
[1] "I PRIMI 10 NUMERI SONO"$ 1&A 1&B A#
[2] 1 A+ &A A 11'4!
[3] A# B A+ &B 2!
[4] $"LA LORO SOMMA VALE " B#
[5]
:
```

4.1.4 - Comando M (modify)

Serve per sostituire una linea di programma con un'altra composta dallo stesso numero di caratteri. Per richiamare la linea desiderata si batte ↑M e i primi due caratteri dopo la parentesi quadra aperta. Ad esempio, M1] per richiamare la linea 1; MIØ per richiamare la linea 10. La linea richiesta viene stampata per intero. Tutto cio' che viene battuto, ovviamente con esclusione dei comandi, va a

rimpiazzare i caratteri della vecchia linea.

Ad una linea possono essere sostituite anche piu' linee, purche' la loro lunghezza (numero dei caratteri) complessiva sia uguale alla lunghezza della linea rimpiazzata.

Battuta la nuova linea si impartisce il comando Q (V.).

4.1.5 - Comando Q (quit)

Nel caso che si voglia modificare solo la prima parte di una linea e' sufficiente riscrivere la nuova linea fino alla modifica (compresa) e impartire poi il comando Q. La parte successiva della linea vecchia viene automaticamente conservata.

ESEMPIO:

```
:M1]
[1] "I PROMI 10 NUMERI SONO"$ 1&A 1&B A#
:[1] "I PRIQ
:L

[1] "I PRIMI 10 NUMERI SONO"$ 1&A 1&B A#
[2] 1 A+ &A A 11'4!
[3] A# B A+ &B 2!
[4] $"LA LORO SOMMA VALE " B#
[5]
:
```

4.1.6 - Comando ← (delete line)

L'effetto di questo comando e' di cancellare completamente la linea che si sta battendo.

4.2 - Le variabili

L'RPN/8 riconosce 16 variabili, identificate da una delle

prime 15 lettere dell'alfabeto inglese e dal carattere "@". Ogni variabile puo' contenere un numero compreso fra Ø e 99.

4.3 - Lo stack

Le operazioni principali vengono eseguite su una struttura a pila (5) che chiameremo nel seguito "stack". Lo stack puo' contenere 12 quantita' numeriche rappresentate in codice BCD (5). Una costante o una variabile puo' essere introdotta nello stack semplicemente facendola seguire dall'operatore di ENTER (V.). Ad esempio, la scrittura

12 A 5

fa si' che lo stack venga a contenere 12,A,5. Il numero 5 si trova nel registro piu' basso dello stack (S1) e 12 in quello piu' alto (S3) (logica lifo).

Le operazioni monadiche vengono di regola effettuate sul contenuto del registro S1 e quelle diadiche sulla coppia S1-S2.

4.4 - Gli operatori

4.4.1 - L'operatore di ENTER

L'operatore di enter e' costituito dal carattere spazio.

Effetto: Il numero o la variabile che precede l'operatore di enter viene posto in S1 e lo stack "sale" di un posto.
Nota: Quando le circostanze lo consentano il carattere spazio puo' essere sostituito da qualunque altro operatore. Ad es.

A B + puo' essere scritto A B+ (eliminando il secondo spazio)

4.4.2 - L'operatore di GO TO (!)

Effetto: Il controllo passa alla linea che ha per etichetta il contenuto di S1 nell'istante in cui viene eseguito il GO TO. Lo stack "scende" di un posto.
Esempio: 3 4+ ! significa: vai alla frase che ha per etichetta 7.
5! " : vai alla frase che ha per etichetta 5.

4.4.3 - L'operatore TEXT (")

Effetto: Tutto cio' che segue l'operatore di TEXT viene direttamente ristampato in fase di esecuzione finche' non si incontra un successivo operatore di TEXT;dopodiche' l'operazione prosegue in modo normale.
Esempio: "PIPPO PLU" "TO ". provoca la stampa di PIPPO PLUTO.

4.4.4 - L'operatore PRINT (#)

Effetto: Il contenuto del registro S1 viene convertito in ASCII e stampato come 1 o 2 cifre. Lo stack "scende" di un posto.
Esempio: 53# stampa 53
4 23# stampa 23
4# 23# stampa 4 23
Nota: Uno spazio viene automaticamente inserito dopo ogni stampa.

4.4.5 - L'operatore DOLLARO ($)

Effetto: Serve per il controllo della stampante e provoca un ritorno carrello/interlinea.
Esempio: "PLUTO" $ "PAPERINO" viene stampata come PLUTO
PAPERINO

4.4.6 - L'operatore END (%)

Effetto: Il controllo viene restituito all'operatore. Deve essere presente in ogni programma per arrestare l'esecuzione e per segnalare la fine fisica del programma stesso.
Nota: L'operatore END non viene ristampato durante il list.

4.4.7 - L'operatore ASSIGN (&)

Effetto: Il contenuto del registro S1 viene assegnato alla variabile che deve di regola seguire l'operatore &. Lo stack "scende" di un posto.
Esempio: 23 &A fa assumere alla variabile A il valore 23
Nota: L'operatore & e la variabile non devono essere divisi da spazi.

4.4.8 - L'operatore IF (')

Effetto: Vengono paragonati i contenuti di S1 ed S2. Se sono uguali si continua la scansione della linea eseguendo tutte le operazioni indicate. Diversamente si procede ad eseguire la linea di programma successiva. Lo stack "scende" di due posti.
Esempio: A B' "PAPERINO"
"PAPERONE"
Se A=B la stampa risulta PAPERINOPAPERONE
SE A≠B la stampa risulta PAPERONE

4.4.9 - L'operatore SUM (+)

Effetto: Vengono sommati i contenuti dei registri S1 ed S2 ed il risultato e' posto in S1. Lo stack "scende" di un posto.
Esempio: 2 3+ # viene stampato 5 (che si trovava in S1 prima della stampa)

4.4.10 L'operatore SEPARAZIONE (,)

Effetto: nessuno
Nota: serve solo per separare tra loro le varie parti di una frase, per maggiore chiarezza tipografica.
Esempio: 3&A, 5&B, A B+ &C

4.5 - Segnalazione degli errori

Diverse condizioni di errore vengono segnalate da una semplice diagnostica. La forma generale della segnalazione e' la seguente:

x? <linea>

dove x e' il codice di errore (V. tab.4.5.1) e <linea> e' la linea alla quale si e' verificato l'errore.

* * * * * * * CODICE DEGLI ERRORI * * * * * * *

CODICE	SIGNIFICATO	NOTE
0	ILLEGAL CHARACTER	CARATTERE ASCII ILLEGALE
1	CHARACTER ERROR	CARATT. IMPROPRIAMENTE USATO
2	PARENTHESIS	PARENTESI NON CHIUSA
3	ILLEGAL LINE NUMBER	CARATTERE NON NUMERICO
4	ILLEGAL COMMAND (*)	
5	OVERFLOW	RISULTATO DI UNA OPER.>99
6	ILLEGAL ASSIGN	ASSEGN. A NON VARIABILE
7	TEXT ERROR	" NON CHIUSO
8	UNKNOWN LABEL	JUMP A ETICHETTA INESISTENTE
9	LINE NOT FOUND (*)	IN RISPOSTA AL COM. MODIFY

(*) INSIEME A QUESTI CODICI VIENE STAMPATA ANCHE UNA LINEA QUALSIASI

Tabella 4.5.1

4.6 - Esempio di programmazione

Si consideri un semplice programma per il calcolo della somma dei primi dieci numeri naturali.

```
* R P N / 8
:C
:[1] "I PRIMI 10 NUMERI NATURALI SONO"$, 1&A, 1&B, A#
:[2] 1 A+ &A, A 11', 4!
:[3] A# ,B A+ &B, 2!
:[4] $ "LA LORO SOMMA E' " B#
:[5]%
:L

[1] "I PRIMI 10 NUMERI NATURALI SONO"$, 1&A, 1&B, A#
[2] 1 A+ &A, A 11', 4!
[3] A#,B A+ &B, 2!
[4] $ "LA LORO SOMMA E' " B#
[5]
:E
I PRIMI 10 NUMERI NATURALI SONO
1 2 3 4 5 6 7 8 9 10
LA LORO SOMMA E' 55
:
```

Seguiamo l'esempio nelle diverse fasi: Si azzera la memoria da eventuali programmi gia' esistenti in essa (comando C); si introduce quindi il programma battendo via via anche i numeri delle frasi tra parentesi quadra. Successivamente si richiede la stampa di una lista del programma per verificare eventuali errori di battitura. Si noti la differenza fra il testo battuto dall'operatore e quello stampato dalla macchina che non e' preceduto, ad ogni riga, dal carattere ":". Infine si passa alla esecuzione (mediante il comando E) e, una volta eseguito il programma, la macchina ristampa ":" in attesa di ordini.

5 - Conclusione

Si e' descritto in linea generale il funzionamento dello

interprete RPN/8 costruito a scopo di studio per funzionare sui micro computers della serie CHILD.

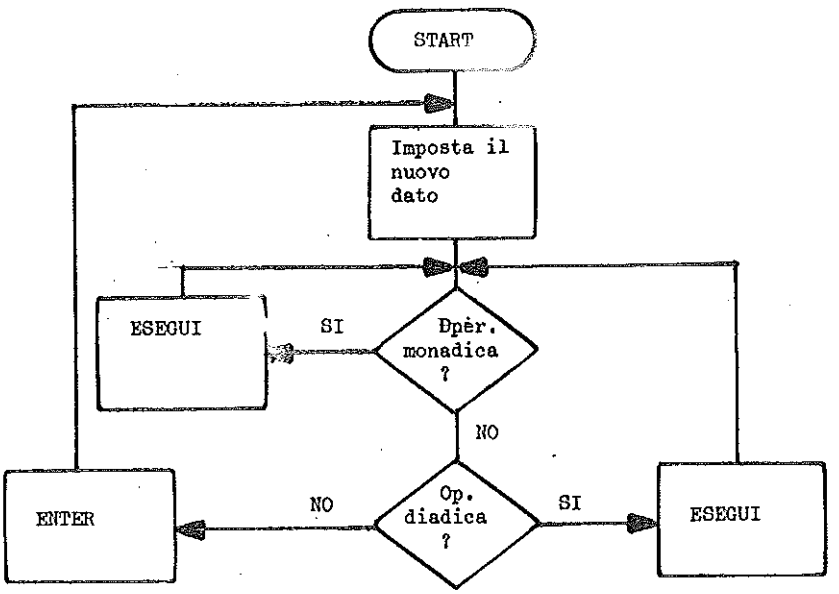
NOTA: I microcomputers CHILD sono fabbricati dalla GENERAL PROCESSOR – Firenze

APPENDICE A
La notazione di Lukasiewicz

Il diagramma di flusso sotto riportato permette di calcolare qualsiasi espressione mediante uno stack (struttura a pila) e la notazione di Lukasiewicz (5,6).

Benche' la soluzione generale richieda una stack illimitato una struttura a 12 registri e' adeguata per la maggior parte dei problemi.

Prima di applicare l'algoritmo e' opportuno scrivere la espressione da calcolare in forma seriale.



APPENDICE B
Grammatica generativa dell'RPN/8

Lo studio teorico riportato nel seguito si riferisce ad una prima versione dell'RPN/8.ed ha puramente un carattere indicativo. Non si e' considerato il programma nel suo insieme ma la singola frase. Segue un esempio di generazione per una frase tipica.
(5,6,7)

Alfabeto terminale: tutti i caratteri appartenenti al set ASCII scriventi
Si ha che $|T_{RPN/8}|=64$ essendo T l'alfabeto terminale.

Alfabeto non terminale: vedi produzioni

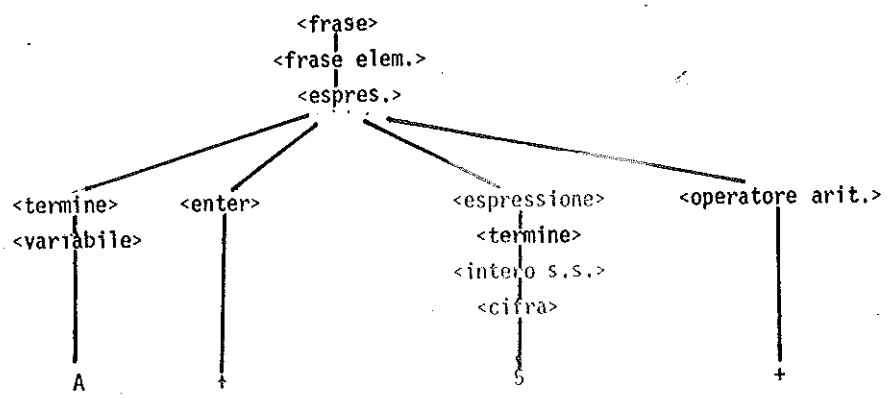
Simbolo distinto: frase

Produzioni:

<frase>::=<frase elementare>|<frase el.><frase>
<frase el.>::=<espressione>|<stampa>|<esp><parola chiave>
<stampa>::=<stampa S1>|<stampa stringa>
<stampa S1>::=<stampa S11>|<stampa S11> <frase>
<stampa S11>::=<esp>#|#
<stampa stringa>::= "<stringa>"
<stringa>::=<carattere>|<stringa><carattere>
<esp>::=<termine>|<termine><enter><esp><operatore aritmetico>|
|<termine><op.arit.>|<op.arit.>
<termine>::=<variabile>|<intero senza segno>
<assegnazione>::=<esp>&<variabile>
<variabile>::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|
<intero senza segno>::=<cifra>|<intero senza segno><cifra>
<carattere>::=<lettera>|<cifra>|<operatore aritmetico>|<delimitatore>|<enter>
<lettera>::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<cifra>::= Ø|1|2|3|4|5|6|7|8|9

<enter>::= ↑| |
<parola chiave>::=# | &
<operatore aritmetico>::= +
<delimitatore>::= "

Esempio:



NOTA: In questa grammatica l'operatore ENTER e' indicato dal carattere "↑".

APPENDICE C

R P N / 8 0 B J E C T L I S T

NOTA : La particolare struttura dei programmi e' dovuta al fatto che al momento della loro stesura non e' stato possibile disporre di un assemblatore. Tutta la programmazione e' stata effettuata direttamente in linguaggio base con l'unico supporto del programma Fair-bug (Debug).

```
M0-50
M0000=28 83 AD 29 02 15 0D 0A
M0008=0A 2A 20 52 20 50 20 4E
M0010=20 2F 20 38 20 00 43 25
M0018=5B 84 04 71 90 3C 16 25
M0020=0D 94 04 72 90 34 21 70
M0028=25 30 84 04 73 90 2B 16
M0030=25 5D 84 08 90 EA 20 20
M0038=51 28 83 E5 90 01 16 21
M0040=7F 53 21 70 25 00 84 F7
M0048=25 20 84 65 25 30 84 11
M0050=25 40 84 0A 25 50 84 BF
?
?M58-100
M0058=70 55 29 02 62 29 01 90
M0060=28 00 7F 90 DA 2A 03 70
M0068=28 83 D6 29 00 3E 26 F6
M0070=45 51 28 83 E5 20 3F 51
M0078=28 83 E5 29 02 26 FF 08
M0080=20 FF 56 36 16 21 70 25
M0088=30 84 F9 46 8E 18 56 28
M0090=02 00 2C 17 2C 09 1C 00
M0098=2C 17 2C 29 00 3E 00 00
M00A0=80 00 80 82 82 82 80 80
M00A8=80 02 80 00 80 80 80 80
M00B0=11 2A 00 C0 43 21 0F 8E
M00B8=8E 8E 0E 10 0D 80 80 80
M00C0=29 00 3E 29 02 B0 29 01
M00C8=40 29 01 20 29 00 68 29
M00D0=02 26 29 01 60 29 02 E0
M00D8=7F 7F 7F 7F 7F 7F 7F 7F
M00E0=FD 29 00 F0 FF 7D FD FD
M00E8=FF FF FF FF FD 29 00 3E
M00F0=2C 20 FE 8E 16 24 66 89
M00F8=92 04 29 01 07 56 20 FE
M0100=8E 46 17 2C 29 00 3E 2C
?
```

```
?M100-150
 M0100=75 55 29 02 62 29 00 3E
 M0110=45 11 55 22 44 90 7F 7F
 M0118=7F 7F 7F 87 7F 0D 7F 7F
 M0120=2C 20 FF 8E 16 56 21 F0
 M0128=84 08 14 24 30 51 28 83
 M0130=E5 46 21 0F 24 30 51 28
 M0138=83 E5 28 01 58 29 00 36
 M0140=16 25 22 94 04 29 00 3E
 M0148=25 0D 84 07 51 28 83 E5
 M0150=90 EF 77 55 29 02 62 2C
?M150-1A0
 M0150=90 EF 77 55 29 02 62 2C
 M0158=20 FF 8E 2C 1C 00 00 00
 M0160=16 53 21 70 25 40 94 17
 M0168=11 2A 01 10 43 21 0F 8E
 M0170=2C 20 FF 8E 16 2C 17 10
 M0178=28 01 57 29 00 3E 76 55
 M0180=29 02 62 96 55 05 02 02
 M0188=02 01 12 22 44 90 11 12
 M0190=11 2A 01 10 43 21 0F 8E
 M0198=16 2C 17 2C 10 29 00 3E
 M01A0=28 83 AD 56 28 83 E5 46
?M1A0-200
 M01A0=28 83 AD 56 23 83 E5 46
 M01A8=25 D1 84 29 25 CC 84 1F
 M01B0=25 C3 84 35 25 C5 84 14
 M01B8=25 4D 84 16 25 5F 84 06
 M01C0=74 55 29 02 62 28 01 F5
 M01C8=29 02 30 29 00 65 29 02
 M01D0=8D 29 02 F6 29 02 26 00
 M01D8=00 00 00 00 00 00 00 00
 M01E0=00 80 80 80 80 60 80 80
 M01E8=2A 03 70 20 90 56 70 17
 M01F0=36 84 1F 90 FB 16 25 0D
 M01F8=94 02 1C 20 FE 8E 90 F6
 M0200=46 21 01 9- 09 16 15 56
?M203-250
 M0208=16 21 0F C6 1C 16 21 0F
 M0210=1C 29 02 26 00 1A 71 50
 M0218=2A 00 06 16 25 00 84 07
 M0220=51 23 83 E5 90 F6 2A 01
 M0228=84 2C 2A 03 FF 28 01 F5
 M0230=28 83 D6 20 3A 51 28 83
 M0238=E5 28 83 AD 21 7F 25 5E
 M0240=84 1E 25 7F 94 0C 20 5C
 M0248=51 28 83 E5 20 FF 8E 84
 M0250=E9 25 0D 94 04 17 90 D9
?
```

```
M258-2A0
M0258=17 7F 28 83 E5 90 DB 29
M0260=01 A0 28 83 D6 45 24 30
M0268=51 28 83 E5 20 3F 51 28
M0270=83 E5 20 FE 8E 28 01 F5
M0278=16 25 0D 94 07 28 83 D6
M0280=29 02 26 51 28 83 E5 90
M0288=F0 00 02 00 00 28 83 D6
M0290=28 83 D6 2A 03 70 16 25
M0298=25 94 04 29 02 26 25 0D
M02A0=94 04 28 83 D6 51 28 83
?M2A8-300
M02A8=E5 90 EC 80 82 80 80 00
M02B0=2A 03 70 16 25 5B 84 0A
M02B8=25 25 94 F8 78 55 29 02
M02C0=62 16 28 00 7F 2C 20 FE
M02C8=8E 16 8D 84 08 20 FF 8E
M02D0=2C 29 02 B3 20 FE 8E 2C
M02D8=28 01 F5 29 00 3E 7F FF
M02E0=2C 20 FE 8E 16 8D 84 08
M02E8=2C 16 25 0D 94 FC 2C 20
M02F0=FE 8E 2C 29 00 3E 28 83
M02F8=AD 21 7F 56 28 83 E5 28
M0300=83 AD 21 7F 57 28 83 E5
?M300-350
M0300=83 AD 21 7F 57 28 83 E5
M0308=28 83 D6 2A 03 70 16 25
M0310=25 94 06 79 55 29 02 62
M0318=25 5B 94 F3 2B 46 8D 94
M0320=EE 47 8D 94 EA 28 01 F5
M0328=16 25 0D 84 07 51 28 83
M0330=E5 90 F6 20 FE 8E 28 01
M0338=F5 29 02 30 28 83 E5 90
M0340=F6 20 FE 8E 28 01 F5 29
M0348=00 80 00 00 00 80 00 00
M0350=00 00 00 30 00 80 00 00
?
```

APPENDIX B – ASCII CODES

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
NULL	00
SOM	01
EOA	02
EOM	03
EOT	04
WRU	05
RU	06
BELL	07
FE	08
H. Tab	09
Line Feed	0A
V. Tab	0B
Form	0C
Return	0D
SO	0E
SI	0F
DCO	10
X-On	11
Tape Aux. On	12
X-Off	13
Tape Aux. Off	14
Error	15
Sync	16
LEM	17
SO	18
S1	19
S2	1A
S3	1B
S4	1C
S5	1D
S6	1E
S7	1F

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
ACK	7C
Alt. Mode	7D
Rubout	7F
!	21
"	22
#	23
$	24
%	25
&	26
'	27
(	28
)	29
*	2A
+	2B
,	2C
-	2D
.	2E
/	2F
:	3A
;	3B
<	3C
=	3D
>	3F
?	3F
[	5B
\	5C
]	5D
↑	5E
←	5F
@	40
blank	20
0	30

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
1	31
2	32
3	33
4	34
5	35
6	36
7	37
8	38
9	39
A	41
B	42
C	43
D	44
E	45
F	46
G	47
H	48
I	49
J	4A
K	4B
L	4C
M	4D
N	4E
O	4F
P	50
Q	51
R	52
S	53
T	54
U	55
V	56
W	57
X	58
Y	59
Z	5A

APPENDICE E

IL SET DI ISTRUZIONI DEL CHILD 8

The F8 instruction set contains over 60 different instructions which may be subdivided into 10 categories: Accumulator, Scratchpad Register, Indirect Scratchpad Address Register, Memory Reference, Data Counter, Status Register, Program Counter, Branch, Interrupt Control and Input/Output instructions. Because 55% of the F8 instructions are only one byte long, programs are short and memory requirements significantly reduced. An alphabetic listing of the instructions is shown below. The following pages contain a complete description of the F8 instructions, including the cycle time. Each cycle is 2 μs for a system with a 2 MHz clock frequency.

F8 ADDRESSING MODES

The F8 instruction set has eight modes of referencing either I/O, CPU registers or bulk memory.

Implied Addressing – The data for this one-byte instruction is implied by the actual instruction. For example, the POP instruction automatically implies that the content of the Program Counter will be set to the value contained in the Stack Register.

Direct Addressing – In these two-byte instructions, the address of the operand is contained in the second byte of the instruction. The Direct Addressing mode is used in the Input/Output class of instructions.

Short Immediate Addressing – Instructions whose addressing mode is Short Immediate have the instruction op code as the first four bits and the operand as the last four bits. They are all one-byte instructions.

Long Immediate Addressing – In these two-byte instructions, the first instruction byte is the op code and the second byte is the 8-bit operand.

Direct Register Addressing – This mode of addressing may be used to directly reference the Scratchpad Registers. By including the register number in the one-byte instruction, 12 of the 64 Scratchpad Registers may be referenced directly.

Indirect Register Addressing – All 64 Scratchpad Registers may be indirectly referenced, using the Indirect Scratchpad Register in the CPU. This 6-bit register, which acts as a pointer to the scratchpad memory, may either be incremented, decremented, or left unchanged while accessing the scratchpad register.

Indirect Memory Addressing – A 16-bit Indirect Address Register, the Data Counter, points to either data or constants in bulk memory. A group of one-byte instructions is provided to manipulate this area of memory. These instructions imply that the Data Counter is pointing to the desired memory byte. The Data Counter is self-incrementing, allowing for an entire data field to be scanned and manipulated without requiring special instructions to increment its content. The memory interface circuit contains two interchangeable data counters.

Relative Addressing – All F8 Branch Instructions use the relative addressing mode. Whenever a branch is taken, the Program Counter is updated by an 8-bit relative address contained in the second byte of the instruction. A branch may extend 128 locations forward or 127 locations back.

ALPHABETIC LIST OF INSTRUCTIONS

ADC	Add Data Counter with Accumulator
AI	Add Immediate with Accumulator
AM	Add Binary Accumulator with Memory
AMD	Add Decimal Accumulator with Memory
AS	Add Binary Accumulator with Scratchpad Register
ASD	Add Decimal Accumulator with Scratchpad Register
BC	Branch on Carry
BF	Branch on False Condition
BM	Branch if Negative
BNC	Branch if no Carry
BNO	Branch if no Overflow
BNZ	Branch if no Zero
BP	Branch if Positive
BR	Absolute Branch
BR7	Branch if ISAR is not 7
BT	Branch on True Condition
BZ	Branch on Zero Condition
CI	Compare Immediate
CLR	Clear Accumulator
CM	Compare with Memory
COM	Complement Accumulator
DCI	Load Data Counter Immediate
DI	Disable Interrupt
DS	Decrement Scratchpad Register
EI	Enable Interrupt
INC	Increment Accumulator
IN	Input
INS	Input Short
JMP	Jump
LI	Load Accumulator Immediate
LIS	Load Accumulator Short
LISL	Load ISAR – Lower 3 Bits
LISU	Load ISAR – Upper 3 Bits
LM	Load Memory
LNK	Link Carry into Accumulator
LR	Load Register (5 Types)
	Scratchpad
	Program Counter
	ISAR
	Status
	Data Counter
NI	Logical AND Accumulator Immediate
NM	Logical AND Memory Accumulator
NOP	No Operation
NS	Logical AND Scratchpad and Accumulator
OI	Logical OR Immediate
OM	Logical OR Memory with Accumulator
OUT	Output
OUTS	Output Short
PI	Push Program Counter into Stack Register Set Program Counter to New Location
PK	Push Program Counter into Stack Register Set Program Counter from Scratchpad
POP	Put Stack Register into Program Counter
SL	Shift Left
SR	Shift Right
XDC	Exchange Data Counters
XI	Exclusive OR Immediate
XM	Exclusive OR Accumulator with Memory
XS	Exclusive OR Accumulator with Scratchpad

ACCUMULATOR GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES	STATUS BITS OVF	ZERO	CRY	SIGN
ADD CARRY	LNK		ACC←(ACC) + CRY	19	1	1	1/0	1/0	1/0	1/0
ADD IMMEDIATE	AI	ii	ACC←(ACC) + H'ii'	24ii	2	2.5	1/0	1/0	1/0	1/0
AND IMMEDIATE	NI	ii	ACC←(ACC) ∧ H'ii'	21ii	2	2.5	0	1/0	0	1/0
CLEAR	CLR		ACC←H'00'	70	1	1	–	–	–	–
COMPARE IMMEDIATE	CI	ii	H'ii' + $\overline{(ACC)}$ + 1	25ii	2	2.5	1/0	1/0	1/0	1/0
COMPLEMENT	COM		ACC←(ACC) ⊕ H'FF'	18	1	1	0	1/0	0	1/0
EXCLUSIVE-OR IMMEDIATE	XI	ii	ACC←(ACC) ⊕ H'ii'	23ii	2	2.5	0	1/0	0	1/0
INCREMENT	INC		ACC←(ACC) + 1	1F	1	1	1/0	1/0	1/0	1/0
LOAD IMMEDIATE	LI	ii	ACC←H'ii'	20ii	2	2.5	–	–	–	–
LOAD IMMEDIATE SHORT	LIS	i	ACC←H'0i'	7i	1	1	–	–	–	–
OR IMMEDIATE	OI	ii	ACC←(ACC) ∨ H'ii'	22ii	2	2.5	0	1/0	0	1/0
SHIFT LEFT ONE	SL	1	SHIFT LEFT 1	13	1	1	0	1/0	0	1/0
SHIFT LEFT FOUR	SL	4	SHIFT LEFT 4	15	1	1	0	1/0	0	1/0
SHIFT RIGHT ONE	SR	1	SHIFT RIGHT 1	12	1	1	0	1/0	0	1
SHIFT RIGHT FOUR	SR	4	SHIFT RIGHT 4	14	1	1	0	1/0	0	1

BRANCH INSTRUCTIONS

In all conditional branches $PC_0 \leftarrow (PC_0) + 2$ if the test condition is not met. Execution is complete in 3.0 cycles.

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES	STATUS BITS OVF	ZERO	CRY	SIGN
BRANCH ON CARRY	BC	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if CRY = 1	82aa	2	3.5	–	–	–	–
BRANCH ON POSITIVE	BP	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if SIGN = 1	81aa	2	3.5	–	–	–	–
BRANCH ON ZERO	BZ	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if ZERO = 1	84aa	2	3.5	–	–	–	–
BRANCH ON TRUE	BT	taa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if any test is true t = TEST CONDITION 2^2 ZERO, 2^1 CRY, 2^0 SIGN	8taa	2	3.5	–	–	–	–
BRANCH IF NEGATIVE	BM	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if SIGN = 0	91aa	2	3.5	–	–	–	–
BRANCH IF NO CARRY	BNC	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if CARRY = 0	92aa	2	3.5	–	–	–	–
BRANCH IF NO OVERFLOW	BNO	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if OVF = 0	98aa	2	3.5	–	–	–	–
BRANCH IF NOT ZERO	BNZ	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if ZERO = 0	94aa	2	3.5	–	–	–	–
BRANCH IF FALSE TEST	BF	taa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if all false test bits t = TEST CONDITION 2^3 OVF, 2^2 ZERO, 2^1 CRY, 2^0 SIGN	9taa	2	3.5	–	–	–	–
BRANCH IF ISAR (LOWER) ≠ 7	BR7	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if ISARL ≠ 7 $PC_0 \leftarrow (PC_0) + 2$ if ISARL = 7	8Faa	2	2.5 2.0	–	–	–	–
BRANCH RELATIVE	BR	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$	90aa	2	3.5	–	–	–	–
JUMP*	JMP	aaaa	$PC_0 \leftarrow H'aaaa'$	29aaaa	3	5.5	–	–	–	–

*Privileged instruction

MEMORY REFERENCE INSTRUCTIONS

In all Memory Reference Instructions, the Data Counter is incremented DC⇐DC + 1.

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES	STATUS BITS OVF	ZERO	CRY	SIGN
ADD BINARY	AM		ACC⇐(ACC) + [(DC)]	88	1	2.5	1/0	1/0	1/0	1/0
ADD DECIMAL	AMD		ACC⇐(ACC) + [(DC)]	89	1	2.5	1/0	1/0	1/0	1/0
AND	NM		ACC⇐(ACC)∧[(DC)]	8A	1	2.5	0	1/0	0	1/0
COMPARE	CM		[(DC)] + $\overline{(ACC)}$ + 1	8D	1	2.5	1/0	1/0	1/0	1/0
EXCLUSIVE OR	XM		ACC⇐(ACC) ⊕ [(DC)]	8C	1	2.5	0	1/0	0	1/0
LOAD	LM		ACC⇐[(DC)]	16	1	2.5	–	–	–	–
LOGICAL OR	OM		ACC⇐(ACC)∨[(DC)]	8B	1	2.5	0	1/0	0	1/0
STORE	ST		(DC)⇐(ACC)	17	1	2.5	–	–	–	–

ADDRESS REGISTER GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES	STATUS BITS OVF	ZERO	CRY	SIGN
ADD to DATA COUNTER	ADC		DC⇐(DC) + (ACC)	8E	1	2.5	–	–	–	–
CALL to SUBROUTINE*	PK		PC_0U⇐(r12); PC_0L⇐(r13); PC_1⇐(PC_0)	0C	1	4	–	–	–	–
CALL to SUBROUTINE IMMEDIATE*	PI	aaaa	PC_1⇐(PC_0); PC_0⇐H'aaaa	28aaaa	3	6.5	–	–	–	–
EXCHANGE DC	XDC		DC_0 ⇄ DC_1	2C	1	2	–	–	–	–
LOAD DATA COUNTER	LR	DC,Q	DCU⇐(r14); DCL⇐(r15)	0F	1	4	–	–	–	–
LOAD DATA COUNTER	LR	DC,H	DCU⇐(r10); DCL⇐(r11)	10	1	4	–	–	–	–
LOAD DC IMMEDIATE	DCI	aaaa	DC⇐H'aaaa'	2Aaaaa	3	6	–	–	–	–
LOAD PROGRAM COUNTER	LR	PO,Q	PC_0U⇐(r14); PC_0L⇐(r15)	0D	1	4	–	–	–	–
LOAD STACK REGISTER	LR	P,K	PC_1U⇐(r12); PC_1L⇐(r13)	09	1	4	–	–	–	–
RETURN FROM SUBROUTINE*	POP		PC_0⇐(PC_1)	1C	1	2	–	–	–	–
STORE DATA COUNTER	LR	Q,DC	r14⇐(DCU); r15⇐(DCL)	0E	1	4	–	–	–	–
STORE DATA COUNTER	LR	H,DC	r10⇐(DCU); r11⇐(DCL)	11	1	4	–	–	–	–
STORE STACK REGISTER	LR	K,P	r12⇐(PC_1U); r13⇐(PC_1L)	08	1	4	–	–	–	–

SCRATCHPAD REGISTER INSTRUCTIONS

(Refer to Scratchpad Addressing Modes)

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES	STATUS BITS OVF	ZERO	CRY	SIGN
ADD BINARY	AS	r	ACC⇐(ACC) + (r)	Cr	1	1	1/0	1/0	1/0	1/0
ADD DECIMAL	ASD	r	ACC⇐(ACC) + (r)	Dr	1	2	1/0	1/0	1/0	1/0
DECREMENT	DS	r	r⇐(r) + H'FF'	3r	1	1.5	1/0	1/0	1/0	1/0
LOAD	LR	A,r	ACC⇐(r)	4r	1	1	–	–	–	–
LOAD	LR	A,KU	ACC⇐(r12)	00	1	1	–	–	–	–
LOAD	LR	A,KL	ACC⇐(r13)	01	1	1	–	–	–	–
LOAD	LR	A,QU	ACC⇐(r14)	02	1	1	–	–	–	–
LOAD	LR	A,QL	ACC⇐(r15)	03	1	1	–	–	–	–
LOAD	LR	r,A	r⇐(ACC)	5r	1	1	–	–	–	–
LOAD	LR	KU,A	r12⇐(ACC)	04	1	1	–	–	–	–
LOAD	LR	KL,A	r13⇐(ACC)	05	1	1	–	–	–	–
LOAD	LR	QU,A	r14⇐(ACC)	06	1	1	–	–	–	–
LOAD	LR	QL,A	r15⇐(ACC)	07	1	1	–	–	–	–
AND	NS	r	ACC⇐(ACC)∧(r)	Fr	1	1	0	1/0	0	1/0
EXCLUSIVE OR	XS	r	ACC⇐(ACC) ⊕ (r)	Er	1	1	0	1/0	0	1/0

*Privileged instruction

MISCELLANEOUS INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES	STATUS BITS OVF	ZERO	CRY	SIGN
DISABLE INTERRUPT	DI		RESET ICB	1A	1	2	–	–	–	–
ENABLE INTERRUPT*	EI		SET ICB	1B	1	2	–	–	–	–
INPUT	IN	as	ACC←(INPUT PORT as)	26as	2	4	0	1/0	0	1/0
INPUT SHORT	INS	a	ACC←(INPUT PORT a)	Aa	1	4***	0	1/0	0	1/0
LOAD ISAR	LR	IS,A	ISAR←(ACC)	0B	1	1	–	–	–	–
LOAD ISAR LOWER	LISL	a	ISARL←a	1101a**	1	1	–	–	–	–
LOAD ISAR UPPER	LISU	a	ISARU←a	01100a**	1	1	–	–	–	–
LOAD STATUS REGISTER*	LR	W,J	W←(r9)	1D	1	2	1/0	1/0	1/0	1/0
NO-OPERATION	NOP		PC_0←(PC_0) + 1	2B	1	1	–	–	–	–
OUTPUT	OUT	as	OUTPUT PORT as←(ACC)	27as	2	4	–	–	–	–
OUTPUT SHORT	OUTS	a	OUTPUT PORT a←(ACC)	Ba	1	4***	–	–	–	–
STORE ISAR	LR	A,IS	ACC←(ISAR)	0A	1	1	–	–	–	–
STORE STATUS REG	LR	J,W	r9←(W)	1E	1	1	–	–	–	–

*Privileged instruction
**3-bit octal digit
***2 machine cycles for CPU ports

NOTES.

Each lower case character represents a Hexadecimal digit
Each cycle equals 4 machine clock periods
Lower case denotes variables specified by programmer

Function Definitions

←	is replaced by
( )	the contents of
(–)	Binary "1"s complement of
+	Arithmetic Add (Binary or Decimal)
⊕	Logical "OR" exclusive
∧	Logical "AND"
∨	Logical "OR" inclusive
H' '	Hexadecimal digit

Register Names

a	Address Variable
A	Accumulator
DC	Data Counter (Indirect Address Register)
DC_0	Data Counter #0 (Indirect Address Register #0)
DC_1	Data Counter #1 (Indirect Address Register #1)
DCL	Least significant 8 bits of Data Counter Addressed
DCU	Most significant 8 bits of Data Counter Addressed
H	Scratchpad Register #10 and #11
I and ii	immediate operand
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISAR	Indirect Scratchpad Address Register
ISARL	Least Significant 3 bits of ISAR
ISARU	Most Significant 3 bits of ISAR
J	Scratchpad Register #9
K	Registers #12 and #13
KL	Register #13
KU	Register #12
PC_0	Program Counter
PC_0L	Least Significant 8 bits of Program Counter
PC_0U	Most Significant 8 bits of Program Counter
PC_1	Stack Register
PC_1L	Least Significant 8 bits of Program Counter
PC_1U	Most Significant 8 bits of Active Stack Register
Q	Registers #14 and #15
QL	Register #15
QU	Register #14
r	Scratchpad Register (any address thru 11)
W	Status Register

Scratchpad Addressing Modes (Machine Code Format)

r = C	(Hexadecimal). Register Addressed by ISAR (Unmodified)
r = D	(Hexadecimal). Register Addressed by ISAR; ISARL Incremented
r = E	(Hexadecimal). Register Addressed by ISAR; ISARL Decremented
r = F	(No operation performed)
r = 0 thru 8	(Hexadecimal). Register 0 thru 11 addressed directly from the Instruction

Status Register

–	No change in condition
1/0	Is set to "1" or "0" depending on conditions
CRY	Carry Flag
OVF	Overflow Flag
SIGN	Sign of Result Flag
ZERO	Zero Flag

APPENDICE F

CARICAMENTO DELL'RPN/8

Per caricare in memoria ed eseguire il programma interpretatore RPN/8 e' necessario seguire la seguente procedura sottoesposta:

1) accendere il microcomputer e la telescrivente.
2) premere RESET con lo switch DEBUG in posizione DEBUG.
3) disporre il nastro nel lettore della telescrivente.
4) battere sulla tastiera della TTY :
   L (CR)
5) porre in START il lettore di nastro.
6) quando la telescrivente stampa "?" porre in FREE il lettore di nastro e riporre il nastro di carta.
7) porre lo switch DEBUG in posizione opposta alla DEBUG
8) premere RESET
9) battere il carattere "spazio" sulla TTY.
10) la TTY stampa la scritta R P N / 8 e si puo' iniziare il lavoro.

- BIBLIOGRAFIA -

1 - A.R. WARD: "LSI Microprocessors and microcomputers: a bibliography" Computer Vol. 7 n.8, luglio 74

2 - G.BECATTINI, C.BOARINO: "Cosa sono i microprocessori" - CQ ELETTRONICA, Aprile-Maggio 1976

3 - G.BECATTINI: "Il CHILD 8/BS" - CQ ELETTRONICA, Giugno, Luglio, Agosto 1976

4 - F.PIRRI, G.BECATTINI: "Child- Un sistema per lo sviluppo e la didattica dei microprocessori", R4-76 Istituto di Elettronica Facolta' di Ingegneria - Giugno 1976

5 - F.CESARINI, R.PINZANI, F.PIPPOLINI: "Fondamenti di sitemi di elaborazione" - ETS, Pisa, 1976

6 - A.ANDRONICO ed altri : "Scienza degli Elaboratori" - Zanichelli, Bologna, 1973

7 - G.AGUZZI, F.CESARINI, R.PINZANI, G.SODA - Introduzione alla programmazione e linguaggio Fortran" - Le Monnier, Firenze, 1973

INDICE

1		Scopi e finalita' dell'RPN/8	pag.	2
2		Sviluppi dell RPN/8	"	2
3		Struttura dell'interprete RPN/8	"	3
	3.1	Il modo di INPUT	"	3
	3.2	Il modo di EXEC	"	6
4		Descrizione operativa dell'RPN/8	"	8
	4.1	Comandi di sistema	"	8
		4.1.1 CLEAR		
		4.1.2 EXECUTE		
		4.1.3 LIST		
		4.1.4 MODIFY		
		4.1.5 QUIT		
		4.1.6 DELETE LINE		
	4.2	Le variabili	"	10
	4.3	Lo stack	"	11
	4.4	Gli operatori	"	11
		4.4.1 ENTER		
		4.4.2 GOTO		
		4.4.3 TEXT		
		4.4.4 PRINT		
		4.4.5 DOLLARO		
		4.4.6 END		
		4.4.7 ASSIGN		
		4.4.8 IF		
		4.4.9 SUM		
		4.4.10 SEPARAZIONE		
	4.5	Segnalazione degli errori	"	14
	4.6	Esempio di programmazione	"	15
5		Conclusione	"	15

APPENDICI:

	A	La notazione polacca inversa	"	17

B	Grammatica generativa dell'RPN/8	pag.	18
C	RPN8 OBJECT LIST	"	20
D	Codice ASCII in notazione esadecim.	"	23
E	Il set di istruzioni del CHILD 8	"	24
F	Caricamento dell'RPN/8	"	28
BIBLIOGRAFIA		"	29

R 4 - 76

CHILD 8

Un sistema per lo sviluppo

e la didattica dei microprocessori

Franco Pirri - Gianni Becattini

- RIASSUNTO -

Vengono dapprima illustrati alcuni dei vantaggi che la tecnologia dei microprocessori apporta al ciclo di sviluppo di un prodotto industriale. Successivamente si descrivono gli elementi e la struttura di un sistema modulare di sviluppo per il microprocessore F8 della FAIRCHILD passando, infine, all'esame dei criteri di approccio a questa nuova tecnologia.

1 - Considerazioni generali.

Sono ormai noti i vantaggi offerti dall'impiego dei microprocessori, o μP, nella realizzazione di sistemi digitali [1,2,3,4] . Ricordiamoli brevemente ponendoli in rapporto alle diverse fasi dello sviluppo industriale di un prodotto.

a - Definizione del prodotto.

L'impiego del μP consente la realizzazione di strutture anche notevolmente complesse a costi molto bassi. Inoltre è relativamente facile ed economico aumentare le prestazioni del sistema e valutarne il costo incrementale.

b - Progetto.

Il progetto consiste prevalentemente nella stesura del software; la conversione in programma del diagramma di flusso è semplice e può essere ulteriormente accelerata con l'uso di assemblatori o compilatori.

c - Realizzazione del prototipo.

Usando una serie di moduli standardizzati è possibile minimizzare il tempo di progettazione e realizzazione dell'hardware, che si riduce così alla semplice costruzione delle interfacce.

d - Messa a punto.

La verifica del prototipo consiste essenzialmente nel debugging dei programmi e la simulazione del sistema su un calcolatore ospite semplifica talvolta questa operazione.

e - Progetto definitivo.

In questa fase vengono disegnati i circuiti stampati finali, le alimentazioni, il contenitore etc.

Molto spesso è conveniente usare direttamente i moduli standardizzati anche nella fase di produzione, garantendo per di più la compatibilità tra i ricambi di apparecchiature diverse. Notiamo anche che il μP consente una grande riduzione dell'hardware con evidenti vantaggi per quanto riguarda il montaggio, la dissipazione di potenza, la documentazione e, in definitiva, il costo.

f - Aggiornamento del prodotto.

Poichè i programmi sono modificabili con una certa facilità è possibile

ovviare all'obsolescenza del prodotto, in modo particolare se si fa uso dei moduli standardizzati.

Per acquisire la necessaria conoscenza delle tecniche di impiego dei μP è opportuno ricorrere a particolari strutture generalmente indicate con il mome di "sistemi di sviluppo" (development systems).

I produttori di ciascun μP mettono sul mercato questi sistemi sia sotto la forma di microcalcolatori completi di periferiche e software, sia come moduli "general purpose".

Un approccio di tipo diverso, che richiede un tempo di approntamento maggiore, ma che garantisce una più profonda comprensione delle funzioni espletate da ogni componente del circuito è quello di realizzare un proprio sistema di sviluppo.

Con l'introduzione del μP il ruolo della programmazione aumenta di importanza, ma conviene mantenere presenti gli aspetti più propriamente circuitali; per questo è auspicabile che all'interno del gruppo di ricerca coesistano esperienze di progettazione hardware e software.

2 - Il sistema CHILD.

Il sistema Child è composto da un certo numero di moduli tra loro compatibili caratterizzati da un costo molto basso e da un buon rapporto prezzo/prestazioni. La struttura dell'insieme è tale da garantire un'ampia possibilità di espansione nel senso desiderato dall'utenze: memoria, I/O, periferiche. Questo rende particolarmente ampio lo spettro di applicazione del sistema CHILD, sia nel campo didattico che in quello dello sviluppo e della produzione di unità a microprocessore.

2.1 - I microcalcolatori CHILD 8.

L'elemento base del sistema CHILD è costituito da un microcalcolatore impiegante il microprocessore F8 prodotto dalla Fairchild [5,6] .

Questo microcalcolatore è stato realizzato in varie versioni sperimentali: la Ø, la 1, la A, la B, la S e la BS.(1) Queste ultime due hanno fornito le prestazioni più interessanti: la S per il costo estremamente ridotto e la BS per la ottima possibilità di espansione. La fig. 1 evidenzia appunto la maggiore flessibilità del modello BS.

Al momento sono stati realizzati tre esemplari del /S mentre due /BS

(1) vedi nota a pag. 12

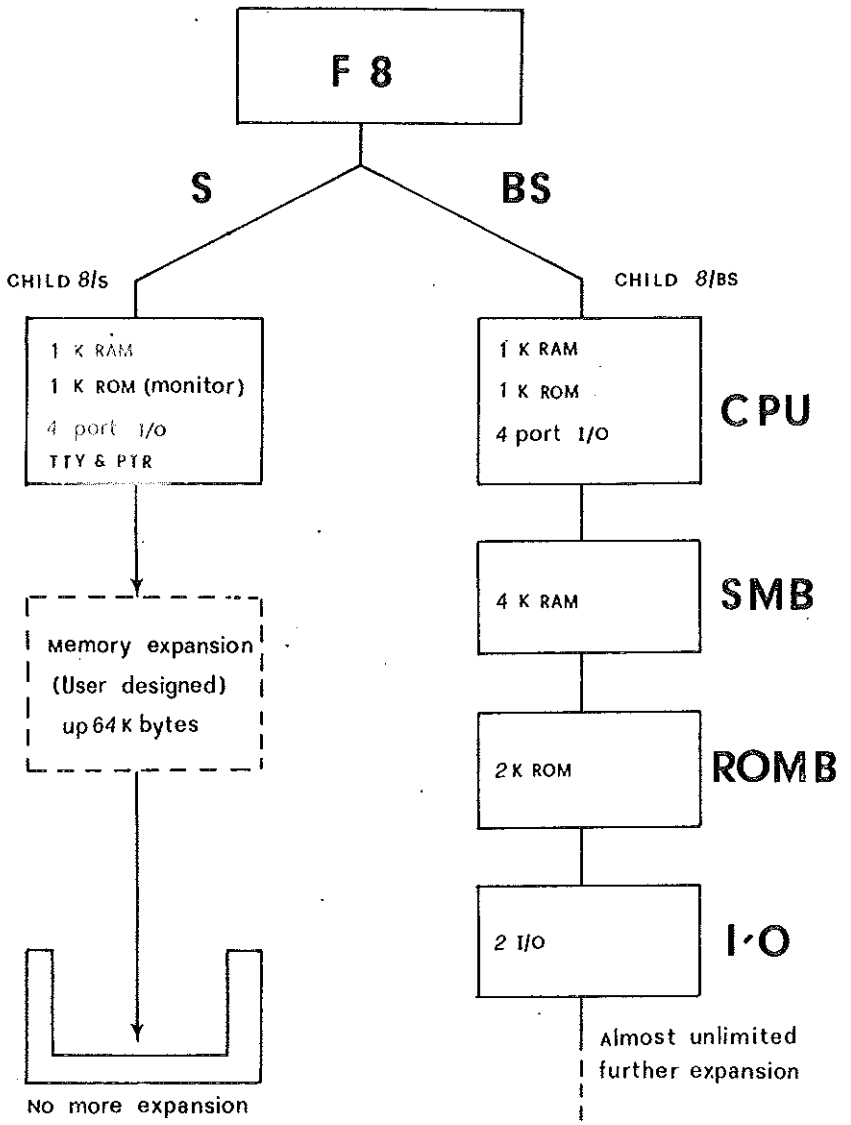


Fig. 1 – Confronto fra i modelli /S e /BS

già montati stanno passando gli ultimi controlli prima del collaudo finale.

In fig. 2 è schematizzata la composizione del CHILD 8/BS, mentre la appendice 1 indica le caratteristiche salienti.

2.2 - Altri elementi della serie CHILD.

Soprattutto in vista della realizzazione di una attrezzatura didattica usabile contemporaneamente da più persone risulta vantaggioso impiegare unità terminali a basso costo. Poichè queste non sono comunemente reperibili in commercio sono state fabbricate con impiego di componenti standard. Esaminiamo brevemente le unità già realizzate ricordando che altri moduli si trovano già allo studio o in avanzata fase sperimentale.

2.2.1 - KB-16.

Si tratta di una tastiera a 16 pulsanti più tasti di controllo. Il set dei caratteri ASCII viene suddiviso in sottoinsiemi di 16 elementi ciascuno, selezionabili con la pressione di un tasto di controllo.

Per quanto l'uso prevalente della codifica esadecimale renda l'operazione sufficientemente facile e spedita è allo studio la possibilità di fare impiego di normali tastiere tipo telescrivente.

All'interno della KB 16 trova posto anche il convertitore parallelo/serie e l'interfaccia a loop di corrente di 20 mA per risultare direttamente intercambiabile con la tastiera della telescrivente meccanica.

2.2.2 - HEXD.

E' un display a 7 segmenti che sotto il controllo di un breve programma (detto HEXMON) permette un completo debugging dello stato della macchina. Il programma HEXMON si trova ancora in fase sperimentale ma una versione provvisoria è già pronta.

2.2.3 - SCA.

Con l'uso delle cassette magnetiche ogni utente può disporre di una grande quantità di memoria di massa a costo molto basso e facilmente maneggiabile.

Per mantenere un ragionevole equilibrio di costi con gli altri elementi del sistema si è optato per un comune mangiacassette musicale tramite il qua

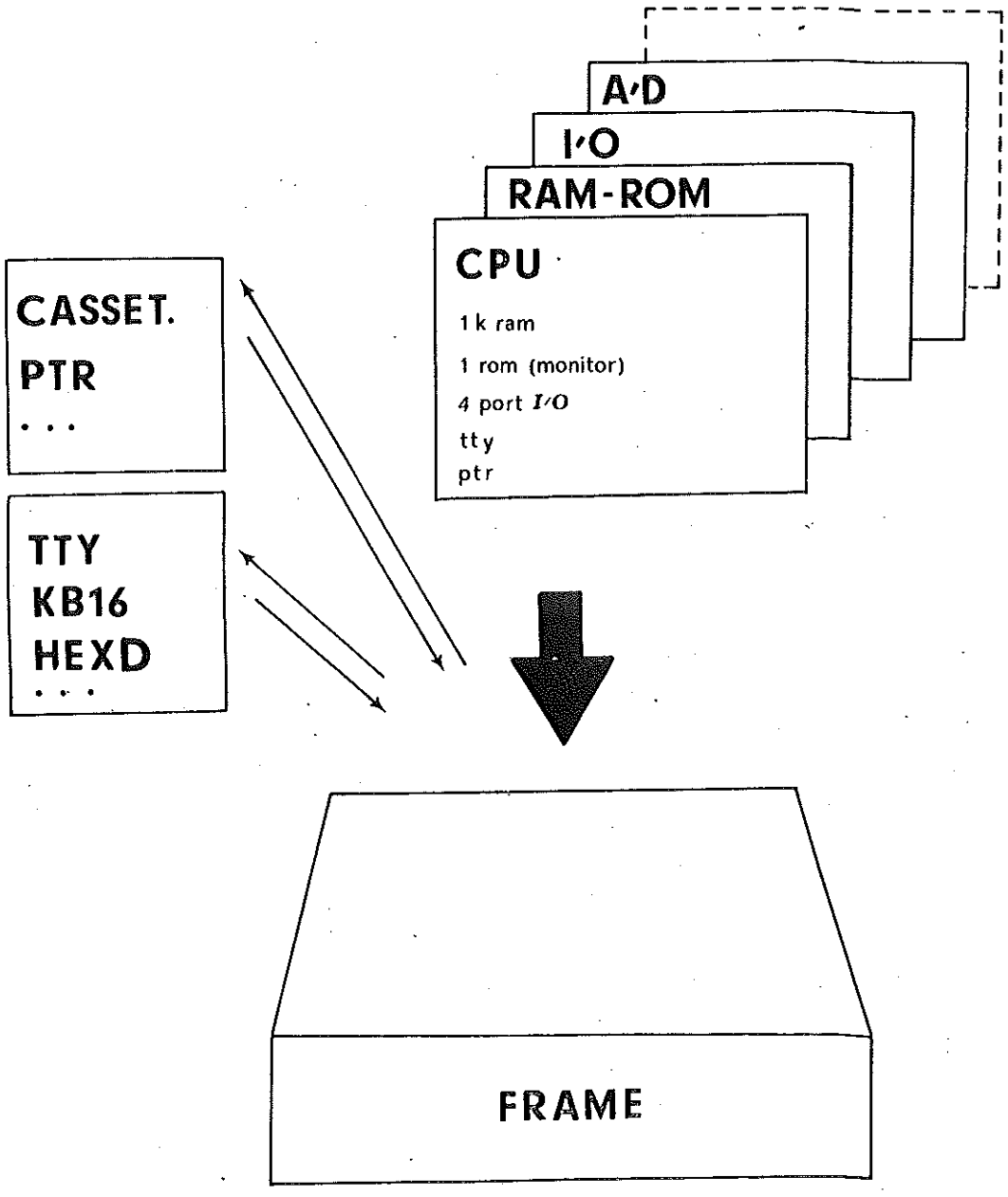


Fig. 2 - Composizione del sistema CHILD

le è possibile registrare programmi e dati su cassette normali fino alla velocità di 500 bit/sec.(2) Velocità maggiori possono essere ottenute con l'impiego di speciali nastri al biossido di cromo.

Lo SCA può essere usato anche per registrare e riprodurre dati analogici con dinamica di ± 1.5 V entro una gamma di 0-250 Hz con buona linearità.

2.3 - Il posto di lavoro.

Il posto di lavoro tipo dispone di un microcalcolatore CHILD 8/BS o /S, con memoria di 1K RAM o più, una tastiera KB 16 con display HEXD e un mangiacassette con SCA (vedi fig. 3). Poichè ogni microcalcolatore CHILD 8 dispone di monitor su memoria ROM [7] è possibile effettuare caricamenti di programmi subito dopo l'accensione degli apparecchi.

Se si organizza il lavoro in modo da evitare la necessità contemporanea da parte di più utenti degli stessi moduli accessori si può limitare il numero di questi ultimi a pochi esemplari senza ridurre l'efficienza del sistema.

3 - Criteri di approccio ai µP.

Un sistema di sviluppo per µP ha due obiettivi: fornire all'utente le cognizioni necessarie per i successivi progetti e servire da base per la costruzione di una vasta gamma di prototipi.

Dal punto di vista didattico possiamo dire che l'uso di un sistema minimo con programmazione in linguaggio macchina costringe l'utente a concentrare la propria attenzione sul significato di ogni operazione e questo si risolve in definitiva in una migliore comprensione del funzionamento dell'elaboratore, almeno finchè ci si limita a programma di qualche decina di istruzioni.

Con questo metodo possono essere scritti i primi sottoprogrammi per la gestione delle operazioni di ingresso-uscita per il governo di periferiche semplici, acquisendo così le cognizioni necessarie per affrontare problemi più complessi.

Non appena si supera il limite di un centinaio di istruzioni diviene necessario l'uso di un linguaggio simbolico. La traduzione in linguaggio macchina può essere effettuata sullo stesso sistema di sviluppo, oppure su un elaboratore ospite. In quest'ultimo caso si parla di cross-assemblaggio o di cross-compilazione e si può scegliere tra un calcolatore proprio od un

(2) vedi nota a pag. 12

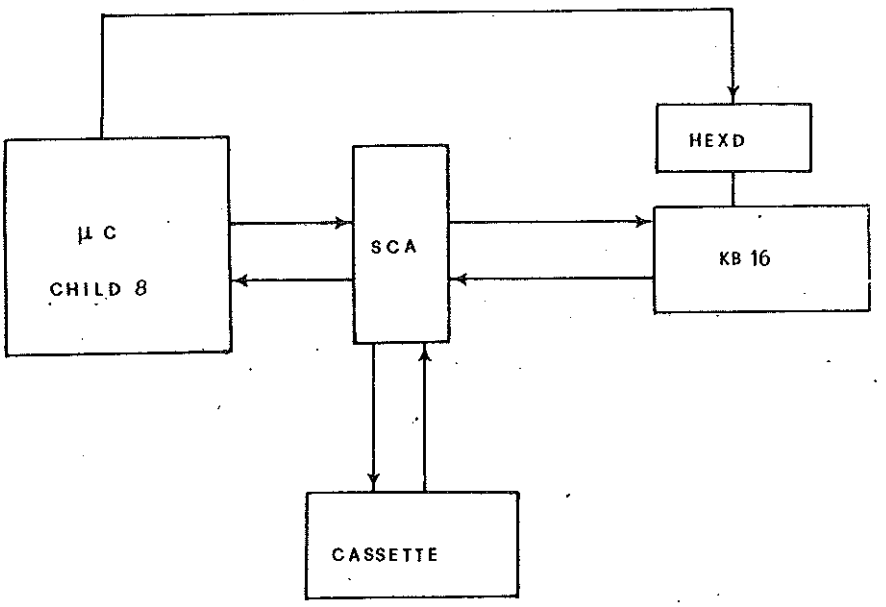


Fig. 3 - Posto di lavoro tipo

terminale time-sharing, ma in ogni caso è bene valutare attentamente i costi di installazione e di gestione del sistema.

Un calcolatore ospite permette la simulazione di un prototipo usufruendo nella stesura dei programmi di tutti i vantaggi derivanti dall'esistenza di un potente sistema operativo, vantaggi che si traducono in un accorciamento dei tempi di sviluppo del software. Bisogna comunque tenere presente che la simulazione non può essere completa e che la fase di sperimentazione sul prototipo non viene quindi eliminata.

Usando il sistema di sviluppo il costo di installazione è modesto e le spese di gestione sono praticamente nulle. A ciò si aggiunge il vantaggio di una costante disponibilità; inoltre è possibile realizzare i moduli più importanti in diversi esemplari, ottenendo così ad un prezzo ridotto la possibilità di suddividere le risorse in più gruppi per lo sviluppo contemporaneo di più prodotti.

Un fattore importante nella valutazione del sistema è il costo, valutato non solo nella fase iniziale, ma tenendo soprattutto conto di quelli che saranno i costi addizionali durante lo sviluppo di altri progetti. E' certamente positivo disporre di un insieme di moduli che consentano di costruire rapidamente sistemi complessi.

Nel caso di piccole o medie serie questo è anche il sistema più economico poichè permette di suddividere il costo dello sviluppo di alcune parti tra più prodotti e riduce il lavoro di documentazione, di immagazzinamento dei pezzi di ricambio, di manutenzione e di istruzione del personale di servizio.

4 - Conclusioni.

Il sistema di sviluppo CHILD è stato realizzato nell'Istituto di Elettronica della Facoltà di Ingegneria di Firenze. Il sistema costituisce un valido mezzo di dimostrazione delle possibilità dei μP e potrà essere usato dal prossimo anno accademico nelle esercitazioni degli studenti.

Un sistema CHILD viene già impiegato per ricerche in campo biomedico ed altre applicazioni sono previste nel campo della trasmissione dati.

Alcuni degli ampliamenti per il sistema base sono già in fase di collaudo e gli sforzi futuri saranno orientati anche verso la preparazione del software di utilità. Per quanto le attuali applicazioni utilizzino il μP F8 non sono escluse ulteriori realizzazioni impieganti altri μP.

Appendice A - Descrizione del microcalcolatore CHILD 8/BS.

Generalità - Il microcalcolatore CHILD 8/BS deriva dai precedenti CHILD da cui si differenzia per una struttura assai più modulare che ne consente la applicazione in un gran numero di sistemi. La caratteristica essenziale dei microcalcolatori della serie CHILD è quella di avere un hardware estremamente ridotto. Tutte quante le funzioni sono infatti espletate dal software. L'innovazione che risulta più evidente anche ad una osservazione superficiale è la quasi assoluta mancanza di comandi sul pannello di controllo. Tale assenza non implica però nessuna rinuncia alle possibilità della struttura più consueta consentendo invece all'utente una facilità di uso assai superiore.

Impieghi - La semplicità della struttura ed il costo ridotto rendono ideale il CHILD 8/BS per applicazione nei controlli di processo. La modularità permette inoltre di espandere il sistema o di adattarlo alle necessità dell'utente con facilità ed economia.

La piastra CPU - La piastra CPU (Central Processing Unit) racchiude gli elementi caratteristici del CHILD 8/BS ed anche da sola costituisce un completo microcalcolatore. In questo modulo trovano posto, oltre al microprocessor FAIRCHILD F8, anche: 1K di ROM (con programma monitor), 1K di RAM, le interfaccie per la telescrivente e per il lettore ottico di banda perforata, 2 livelli di interrupt, 32 bit bidirezionali di I/O.

L'inserzione della CPU in un bus non dedicato consente l'espansione della memoria ed il collegamento di numerosi accessori (stampanti, cassette, etc.).

Caratteristiche principali del CHILD 8/BS - Parola di 8 bit/Ciclo di macchina di 2 µs/Set di 70 istruzioni /1 Accumulatore/64 registri/2 registri indice/Altri registri per status, subroutine etc./4 ports bidirezionali di I/O/2 livelli di interrupt/2 timers programmabili per applicazioni in tempo reale/Power-on reset/Espandibilità della memoria fino a 64 K bytes con RAM, ROM, PROM, EPROM/Espandibilità dei port di I/O praticamente illimitata/Monitor su ROM.

Pannello di controllo software - In qualunque momento si può passare sotto il controllo del "pannello software" con la semplice pressione di un tasto. Tutte le funzioni, e molte altre ancora, eseguibili da un comune pannello

con switches e leds si possono eseguire da telescrivente: lettura o modifica della memoria o dei registri (in esadecimale), caricamento o perforazione di nastri di carta, caricamento o registrazione di nastri magnetici, comando di "start" da qualunque locazione di memoria etc.

Lo status della macchina viene automaticamente salvato quando si richiede di passare sotto il controllo del "pannello software".

Altre caratteristiche - Oltre al "pannello software" il monitor fornisce all'utente 7 sottoprogrammi per la gestione delle operazioni di I/O. Ogni piastra preleva la sua alimentazione da una unica linea a 16 Vcc ed effettua tutte le regolazioni sulla piastra stessa.

Le interconnessioni fra piastre diverse avvengono tramite un bus "non dedicato"; questo significa che l'espansione può essere fatta indipendentemente in senso di memoria o di I/O o di entrambe, potendo ogni piastra essere inserita in qualunque posizione.

Componenti usati - I principali circuiti integrati usati nel microcalcolatore sono:

3850	Central processing unit (CPU)
3851A	Program storage unit (PSU)
3853	Static memory interface (SMI)
2102-2	Memoria statica 1Kbit

La logica accessoria è stata realizzata, ove possibile, con circuiti C-MOS e TTL-LS per minimizzare la dissipazione di potenza. L'alimentazione è stabilizzata sulla piastra stessa tramite regolatori a tre terminali. La tensione di 5 V è fornita dall'integrato 78HD5, capace di erogare una corrente di 5 A, e può essere usata per alimentare anche piccoli moduli esterni.

NOTE

(1) - Le versioni Ø, 1, A e B del microcalcolatore CHILD 8 sono state da noi progettate utilizzando alcuni circuiti integrati della serie F 8 della FAIRCHILD. Queste versioni sono state realizzate in esemplare unico e rivestono interesse solo per il fatto che il lavoro di progettazione ci ha contentito di acquisire la necessaria esperienza per il progetto della versione BS.

Il CHILD 8/S è l'"F 8 evaluation Kit number one" della FAIRCHILD a cui sono state aggiunte le alimentazioni, il contenitore etc.

Il modello BS usa anch'esso gli integrati della serie F 8 ma è realizzato in modo da poter essere il modulò principale di un sistema espandibile a piacere su un bus non dedicato (cfr. appendice A).

(2) - Lo "Standard Cassette Adapter" (SCA) da noi realizzato elabora un segnale analogico modulato in frequenza compatibile con i normali registratori.

Attualmente l'adattatore lavora alla velocità di 110 band perchè si è ritenuto più conveniente mantenere una diretta compatibilità con la telescrivente ASR 33.

BIBLIOGRAFIA

1 - Computer - Vol. 7 n. 7, giugno 1974 - Numero speciale dedicato all'architettura dei microprocessori.

2 - A.R. WARD: "LSI Microprocessor and microcomputers: a bibliography" - Computer Vol. 7 n. 8, luglio 1974.

3 - Alta Frequenza - Vol. XLIV n. 6, giugno 1975 - Fascicolo monografico dedicato ai microelaboratori.

4 - Electronics - Vol. 49 n. 8, 15 aprile 1976 - Numero speciale dedicato ai microprocessori.

5 - "Preliminary F8 microprocessor data book" - FAIRCHILD SEMICONDUCTOR, 27 giugno 1975.

6 - "A guide to programming the FAIRCHILD F-8 microcomputer" - FAIRCHILD SEMICONDUCTOR, 27 giugno 1975.

7 - "User's manual F8 design evaluation kit number one" - FAIRCHILD SEMICONDUCTOR, 27 giugno 1975.